(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 May 2001 (10.05.2001)

PCT

(10) International Publication Number
WO 01/33433 A1

(51) International Patent Classification[7]: G06F 17/30, 7/00, 17/00

(21) International Application Number: PCT/US00/30020

(22) International Filing Date: 31 October 2000 (31.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/430,833 1 November 1999 (01.11.1999) US

(71) Applicant: XCODERS INC. [US/US]; Corporation Trust Center, 1209 Orange Street, Wilmington, DE 19801-1120 (US).

(72) Inventor: ANNER, Amotz; 46 Jerusalem Street, Flat 9B, 44369 Kfar-Saba (IL).

(74) Agent: BERUBE, Robert, B.; Marsh Fischmann & Breyfogle, LLP, 3151 South Vaughn Way, Suite 411, Aurora, CO 80014 (US).

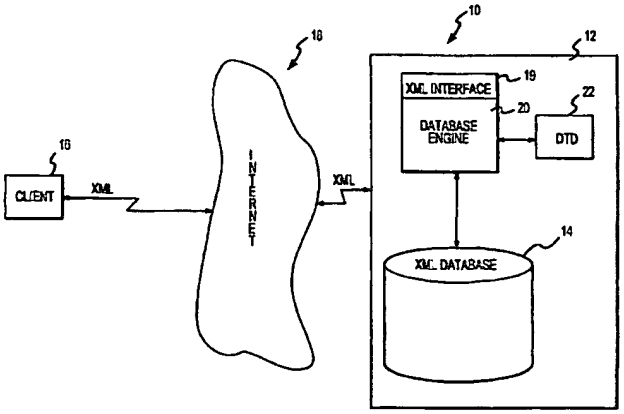
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND APPARATUS FOR ESTABLISHING AND USING AN XML DATABASE



(57) Abstract: A server provides access to an XML database, one embodiment, an XML database (14) is stored on a server website (12) accessible by a client (16) via the internet (18). The client (16) can access the website (12) to establish the XML database (14), populate the database (14) with XML documents, and retrieve data from the database (14). All of these functions are executed via a database engine (20) of the website (12). The illustrated engine (20) is associated with an XML interface module (19) for communicating with XML compatible browsers and receiving XML communications related to establishing, accessing and managing the database (14). The XML database (14) is defined by registering an indexing definition such as a DTD (22) in the database engine (20). The invention allows XML documents to be parsed and stored in a relational database together with indexing information for each portion of the document content.

WO 01/33433 A1

METHOD AND APPARATUS FOR ESTABLISHING AND USING AN XML DATABASE

Field of the Invention

The present invention relates in general to databases that can be accessed via a network such as the internet and, in particular, to a database structure that can be accessed using an Extensible Markup Language (XML) interface and that can store XML documents.

Background of the Invention

A number of database management systems are currently available including systems from Microsoft, Sybase, Informix, Oracle and others. Such database management systems generally use a tool such as the Structural Query Language (SQL) to allow users to define, manipulate and access data in a database. The resulting databases are generally relational databases where data is stored in tables. The information stored in a particular table is typically narrowly limited in subject matter, e.g., to information about a single "entity." Thus, for example, an order management database may include one table of customer names, one table of orders, one table of order prices, etc. A user can retrieve information from the database by entering (or causing a search tool to enter) a query such as a select statement. Based on the query, information from various tables may be joined using certain relational operators. An important advantage of such database structures is that the information can be accessed and organized in many different ways depending on the nature of the query entered by the user.

Such databases are not directly accessible via the Internet. In order to access such a database, a page creator must include a URL specifying a CGI script. Upon the user clicking on such a link, the server will run a process that will execute the script and generate some output that is sent back to the user's browser. Scripts are dynamic programs for performing specialized functions. These scripts are generally stored on the server.

Communications between the client and server and the documents or objects stored in the database and provided to the client often utilize the Hypertext Markup

Language (HTML), which is a pervasive data format for the World Wide Web. HTML provides a simple mechanism for formatting documents and transmitting them over the Web. An emerging standard is the Extensible Markup Language (XML). Both HTML and XML use tags for document handling. However, HTML focuses primarily on document formatting or layout whereas XML focuses more on content. In particular, HTML utilizes a fixed set of tags that primarily specify format. The XML tags, by contrast, delimit segments of data but allow the application to define the interpretation of the tags. As a result, XML has certain advantages over HTML relating to data reuse and interchange.

Current databases and database tools are subject to a number of limitations. First, such databases and tools have a limited ability to manage complex data structures. Specifically, for applications where the number of tables in a relational database becomes large, the response time for queries may become very slow. In addition, as current database engines are not directly accessible via the web, such access is dependent on scripts resident at the server to manage access to the database. Such scripts are dynamic programs that may malfunction and require maintenance by the operator of the server. Moreover, such scripts limit the customizability of database applications (e.g., the ability to accommodate applications customized for each user), and allowing wide access to such scripts may adversely impact server security. The use of scripted database tools also inhibits direct access to stored documents. Data reuse and interchange as between databases or documents is also limited by current database structures. Finally, current database structures and tools are generally not adapted to provide an efficient XML interface or to facilitate storage of XML documents.

Summary of the Invention

The present invention is directed to a method and apparatus for use in establishing and accessing a database that can store documents or objects in XML and that can be accessed via an XML interface. The invention allows for indexing parent-child relationships so that complex queries can be performed without scanning the content of the stored documents. As a result, the invention supports a number of advantages including: document storage for a large number of XML documents; content addressing

for stored documents or portions thereof for simple and secure location of documents based on content without resort to server side scripts; data extraction for reuse in other XML documents; data interchange with little or no coding for generating and processing data interchange messages; support for complex, evolving and one-of-a-kind databases with a minimum of coding or no coding at all; and support for complex applications that are entirely browser based and that can be customized for each individual user without degrading system reliability.

According to one aspect of the present invention, a method and corresponding apparatus are provided for storing and accessing information in a network. The method involves establishing a database of information objects at a server node accessible via the network and establishing an XML interface in connection with the server node. The XML interface can be used for accessing the database based on XML communications from a client such as communications including an XPATH statement. The method further involves receiving an XML communication from a client node and using the XML interface for storing or retrieving one or more information objects or portions thereof relative to the database. In this manner, the database can be used to store and access XML documents.

The step of establishing the XML interface preferably involves registration of a Document Type Definition (DTD) or XML Schema with the database server. The DTD must define, inter alia, which tags may be used, how they may be nested and what data is acceptable for each tag. Once the DTD is registered with the server database, any XML document that complies with the DTD is deemed valid and can be added to the database and retrieved from it. It will thus be appreciated that the invention supports complex applications that can be highly customized, including applications customized on a user-by-user basis.

In accordance with another aspect of the present invention, a method and corresponding apparatus are provided for handling information queries embedded in an address request. The method includes the steps of: establishing a database of information objects accessible via a network having an internet addressing system; receiving an address request that includes a first portion identifying a network address of the server, e.g., an URL, and a second portion including an information query with coded

instructions for searching the database; and executing the coded instructions to access one or more of the information objects of the database or a portion or portions thereof. The second portion of the address request may further include a database identifier.

The information query embedded in the address request may be, for example, a document ordinal or a logical XPATH statement as referenced above, and is referred to herein as a Universal Resource Identifier (URI). An address request in accordance with the present invention may therefore, for example, be of the form "www.[server].com/[database]/[URI]." The server can then directly access the database based on the URI. The invention thus supports database queries without the intermediate step of accessing a conventional server side search tool and without involving server side scripts that may malfunction, require maintenance, adversely impact server security and limit customizability.

According to another aspect of the present invention, a method and corresponding apparatus are provided for accessing a database based on parent-child relationships of stored information objects so as to support complex queries and allow for timely database access. The method includes the steps of: establishing a database of information objects including at least one parent object and at least one child object; defining a Boolean matrix reflecting parent and child relationships as between information objects of the database; and, given information regarding a first one of the child object and parent object, executing a transform operation using the Boolean matrix to identify the other one of the child object and parent object. The parent object is associated with a parent tag system (e.g., defined by opening and closing angle brackets) and the child object is associated with a child tag system where the child tag system is incorporated within the parent tag system. As will be described in more detail below, a transform operation can be performed on the Boolean matrix to identify all child data nodes of a set of parent data nodes. The operation can be repeated to identify grandchildren, great grandchildren etc. An inverse operation generates a vector of parent nodes given a vector of child nodes. Again, the inverse operation can be repeated to identify further ancestry relationships. These parent-child relationships can be used to support complex queries and to allow for timely accessing of information in applications involving complex data structures.

In accordance with a further aspect of the present invention, a method and corresponding apparatus are provided for mapping XML documents into a relational database. In this regard, a process for storing an XML document comprises the steps of: receiving an XML document; parsing the XML document to create a number of information objects; and mapping the XML document into a relational database by establishing, for each one of the information objects, a database entry including information content of the information object and indexing information for use in identifying the information object. By virtue of such mapping, portions of an XML document can be identified based on content without scanning the content of the document.

A method for accessing an XML document stored in such database comprises the steps of: accessing a database structure storing XML documents where a particular XML document is parsed into separate information objects and said objects are mapped into a relational database; causing an algorithm to be executed to retrieve the separate information objects of the XML document and to reconstruct the XML document or a portion or portions thereof from the information objects; and receiving the reconstructed XML document or portion or portions thereof from the database structure. This process for accessing the database may make use of parent-child relationships and associated Boolean transform operations as discussed above so as to quickly reconstruct an XML document.

Brief Description of the Drawings

For a more complete understanding of the present invention and further advantages thereof, reference is now made to the following detailed description taken in conjunction with the drawings, in which:

Fig. 1 is a schematic diagram of a network in which the present invention may be implemented;

Fig. 2 is a flowchart illustrating a process for establishing and using an XML database in accordance with the present invention; and

Fig. 3 is a flowchart illustrating a process for administering an XML database in accordance with the present invention.

Detailed Description

In the following description, the invention is set forth in the exemplary context of a system for storing, indexing and retrieving XML documents in a database resident on an internet server. In particular, the following description is divided into the following sections: 1) establishing and maintaining an XML database; 2) XML interface specifications; 3) indexing parent-child and other family relationships; and 4) mapping XML documents to a relational database.

1. Establishing and Maintaining an XML Database

a. Database Definition

Fig. 1 is a schematic diagram of a database system 10 in accordance with the present invention. In the illustrated embodiment, an XML database 14 is stored on a server website 12 accessible by a client 16 via the internet 18. The website 12 may be, for example, an HTTP 1.1 server website. In accordance with the present invention, the client 16 can access the website 12 to establish the XML database 14, populate the database 14 with XML documents, and retrieve data from the database 14. All of these functions are executed via a database engine 20 of the website 12. As will be understood from the description below, communications between the client 16 and the database engine 20 include XML statements. The illustrated engine 20 is associated with an XML interface module 19 for communicating with XML compatible browsers and receiving XML communications related to establishing, accessing and managing the database 14. Although the invention is thus illustrated with respect to an internet based example, it will be appreciated that various aspects of the invention are more broadly applicable in a variety of networks and database systems.

The XML database 14 is defined by registering an indexing definition in the database engine 20. In the illustrated embodiment, the definition includes a Document Type Definition (DTD) 22 as generally set forth in the XML standard, but the definition may include an XML Schema or other element providing similar definitional information. The DTD 22 establishes syntax to define a document type for the XML database 14. The illustrated DTD 22 defines which tags may be used, how they may be nested, what data is acceptable for each tag, and any other information that is relevant for

a particular application. An XML document that complies with the DTD 22 is deemed to be valid and can be added to the XML database 14 and retrieved from it.

The concepts of tags and family relationships, which are defined in the DTD 22, are important to understanding preferred implementations of the present invention. In XML, tags are used to delimit data segments that may be defined by a processing application. In particular, data segments are delimited by a set of starting angle brackets (<...>) and a set of ending angle brackets (</...>). Within each bracket set is an identifier for the tag, and between the tag sets is a tag content. Tags can be nested to define arbitrarily complex data structures The following is a simple example showing the use of a tag statement to convey the name "John Doe" which may be added to or retrieved from a name database table such as a customer name table:

<name> <family> Doe </family > <first> John </first > </name>

It will be observed that this tag statement uses the tags "name," "family" and "first." Moreover, the tags "family" and "first" are nested within the tag "name." The tag "name" is thus a parent of the tags "family," and "first" which are children in this example. More complicated data structures can be defined including grandparents, great grandparents etc. and grandchildren, great grandchildren etc. Such lineage or ancestry is used in accordance with the present invention as described below to facilitate storing, searching and retrieving of XML documents. The DTD 22 stores these tags, the family relationships between the tags, and data type of the content, and other information as relevant to an application. The database engine 20 may provide a user friendly set of menus, screens and the like for entering this DTD information.

Once the DTD is entered, the database engine 20 creates an enumeration of all tags in the DTD and for each tag the enumeration of possible attributes. Such attributes are permitted within the XML standard and have many possible uses. In the context of the example above, an attribute may be assigned to the tag "name" in order to include the honorific title "sir" as follows:

<name honorific = "sir">

Alternatively, such an honorific title may have been provided by way of a further child tag. The enumeration of tags, attributes and relationships is made available to clients for ease of understanding and using the database. It will be appreciated that this flexibility

in defining and registering a DTD allows for substantially unlimited customizability of databases and supports applications customized for each user.

The database engine 20 associates a permanent ordinal with each XML document instance in the database starting from 1. Upon adding a valid XML document instance, the database engine 20 will create an enumeration of its nodes. Concatenating the document ordinal and the node number will create a database wide unique node ID. In addition, the database engine 20 creates a DTD with the view of the whole database as a single XML document associated with the ordinal 0. This allows for convenient identification of the overall database.

b. Database Indexing

As will be discussed in more detail below, XML documents may be parsed into segmented portions for storage in the database. It is useful to index the documents and segmented portions for efficient identification, access and reconstruction. In this regard, for each unique combination of tag number and tag contents, the database engine 20 maintains a Boolean vector of node ID's, where bit n is set if and only if node n contains the tag-contents combination. The engine 20 also maintains a Boolean vector of tag numbers regardless of contents (index of ALL), as well as an index of all roots. This Boolean vector is compressed in such a manner that there is an easy way to extract from it a Boolean vector of document ordinals. It will be appreciated that such a vector of document ordinals is useful in document searching and reconstruction.

c. Database Growth

Database growth may be evolutionary or revolutionary. In evolutionary growth, an enlarged database definition is provided by a superset DTD that is a strict enlargement of a base DTD. That is, a superset DTD is produced by adding additional definitions to an existing DTD, without any deletions and without re-ordering tags or changing the nesting structure of existing tags. In the context of the example provided above, the name database may be evolved to include a field for entering middle name information. In this regard, a new tag "middle_name" may be defined as a child of the tag "name." It will be appreciated that this additional definition does not affect the existing tags or their nesting structure. The database engine 20 will support the replacement of a base DTD by a DTD that is a superset of it. In particular, the engine 20 will maintain the enumeration of the

existing tags, and will extend it to enumerate the new tags. Existing documents will not need to be changed.

The case of revolutionary growth is more complicated. Revolutionary growth involves replacing a DTD with a DTD that is not a superset of it. For example, in the context of the example noted above, it may be desired to redefine the "family" tag as a parent of "first" in order to support a desired nesting structure for multiple members of the same family. The database engine 20 can support such a redefinition of the DTD 22 by using a transformation module. For example, the transformation module may be an XSLT module or a module providing similar functionality. This module is used to transform the existing documents in the XML database 14 into the new format.

2. XML Database Interface Specifications

As noted above, the database engine 20 may function as a HTTP 1.1 server. The engine 20 supports client access to DTD definitions, individual XML documents, all documents in the database (using the ordinal 0) and external support files. An important aspect of the present invention is that such access can be executed via an XML interface. In particular, the client can embed an XML query or URI within an address request transmitted across the Internet 18. For example, such an address request may have the form:

server_domain_name/database_name/XML_document_ordinal

where server_domain_name is the URL of the website 12, database_name is an identifier of the XML database 14 and XML_document_ordinal is an identifier or URI for an XML document or portion or portions thereof to be accessed. Such a URI may be used to access the DTD definitions, individual XML documents or a portion or portions thereof, or all documents in the database. Where all documents in the database are identified, the URI may further include an XPATH statement to invoke XPATH logic for selecting a portion or portions of the database that will be returned to the client as a single XML document. In this regard, for example, a customized XML document may be generated including only titles or other summary information for some portions of the original XML document and the full content of other portions.

A complete interface to the illustrated XML database 14 is defined using XML only. Programs, and in particular, scripting languages such as Perl, are able to read, write, query and update the database by creating TCP/IP connections to a predefined port number on the server and by sending and receiving properly formatted XML documents over these connections. It will be appreciated that the XML database 14 can thus be accessed directly without invoking any server side scripts. The invention thus avoids the potential problems of script malfunctions, server side script maintenance and security concerns.

The server site 12 may also provide a standard binary API. This will support accessing the XML database 14 using TCP/IP and a port number different from the XML based service. The primary advantage of this access mode will be the ability to read XML documents in the internal representation, so that they do not need to be parsed by the receiving application.

3. Indexing Family Relationships

As noted above, the family relationships between the tags may be useful in searching the XML database 14 and reconstructing parsed documents from the XML database 14. That is, given an identified child node, it may be desired to identify all or certain parent, grandparent, great grandparent etc. nodes thereof. Conversely, given a parent node identifier or a set of such identifiers, it may be desired to identify all or certain children, grandchildren, great grandchildren, etc. This can be accomplished by defining a Boolean matrix that reflects family relationships and performing certain transform operations on the Boolean matrix to identify specified family members. The following is an algorithm for identifying child nodes starting from a given parent node.

1. Let a_i (i=0,n) be the set of all nodes in an XML document (or database).
2. Define a Boolean matrix M of rank n such that $M_{ij} = 1$ if and only if node j is an immediate child of node i.
3. Let V^s be a Boolean vector representing a subset s of a_i such that V^s_i is 1 if and only if node i is contained in the subset s.
4. Define the Boolean matrix transform operation $V^s M = V^c$ as: $V^c_j = OR_{i=0..n} V^s_i$ AND M_{ij}.

5. Then V^c is exactly the set of all nodes that are a child of a least one node in a_i.

Thus, this algorithm identifies all direct children of the given parents. This transform operation can be repeated twice to identify grandchildren, thrice to identify great grandchildren, and so on. A generalized descendant relationship can be computed by repeatedly transforming the original vector and ORing the intermediate results until a null vector is generated. Termination is guaranteed by the well-formedness requirement of XML that assures proper nesting such that cyclic relations cannot exist. That is, the relationship graph is hierarchical such that a descendant of any node is not an ancestor of the same node. Accordingly, the process noted above will not result in an infinite loop. The inverse operation (MV^s) generates a vector of parent nodes. This inverse operation can be repeated as required and generalized to an ancestor relationship as noted above in connection with descendant relationships.

The use of a transformation matrix to identify specified portions of an XML document is illustrated by the following example. For the purposes of this example, it is assumed that the XML document is divided into chapters identified by the tag "chapter." One or more of the chapters includes a title identified by the tag "title," where the title tag is a direct child of the chapter tag.

Using XPATH notation, and assuming that the starting context is a database, the XPATH statement: "child::node()[child::chapter[child::title='Introduction']]" should locate all <title> nodes with the content "Introduction," that are children of a <chapter> node, one level below the root of each document. The processing steps are as follows:

1. Start with the Boolean vector for document roots in a database V^{root}.
2. Compute $V^{top\text{-}child} = V^{root}$ M, yielding all second level nodes.
3. Retrieve from index $V^{chapter}$, the Boolean vector of all "chapter" nodes, regardless of contents.
4. Compute $V^{top\text{-}chapter} = V^{chapter}$ AND $V^{top\text{-}child}$, the Boolean vector of all second level <chapter> nodes.
5. Compute $V^{tc\text{-}child} = V^{top\text{-}chapter}$ M, The Boolean vector of all children of a second level <chapter> node.
6. Retrieve from index V^{intro}, the Boolean vector of all <title> nodes with contents "Introduction."

7. Compute final result as $V^{result} = V^{tc\text{-}child}$ AND V^{intro} all third-level <title> nodes with the contents "Introduction," that are children of a second-level <chapter> node.

V^{result} can now be used to retrieve all the required nodes from document storage. By truncating V^{result} to the document level, all documents that contain such nodes can be retrieved. It will thus be appreciated that the transform matrix provides an efficient tool for identifying documents or portions thereof based on indexed family relationships without the need to scan the entire content of the document.

4. Mapping XML Documents to a Relational Database

The model for mapping an XML document into a relational database is the "grove of groves" model. That is, the content of each node is a linear array of PCTEXT items, possibly intermixed with contained nodes, each having the same type of structure. An initial step in the mapping process involves enumerating tags, attributes, nodes, documents and groves. All the possible tags to be defined in a DTD are enumerated, starting from 1. Similarly, all the possible attributes for any defined tag are independently enumerated, starting from 1. When a DTD is expanded, the enumeration is likewise expanded, the ordering of the enumeration is immaterial and it need not be dense.

Within each document, the nodes are enumerated starting from 0, which is reserved for the root node (i.e., to identify the entire document). The enumeration of nodes is in the order of encountering the nodes when parsing the document, so it will always be dense. The documents in a database are enumerated in chronological order of addition to the database, starting from 1, and when documents are deleted, holes will be left in the enumeration. All the members of a grove are densely enumerated, starting from 1.

As noted above, parent-child relationships are also important in accessing a stored document or portion thereof and in reconstructing the document. These relationships are encoded in a transform matrix as described above.

The mapping results in a single table for each content portion having the following columns:

#	Column Name	Comment
1	Document number	Primary index
2	Node number	
3	Tag number (or tag name)	
4	Grove ordinal	0 if attribute
5	Attribute ordinal	0 if content
6	Placeholder flag	TRUE to give position in external grove
7	Content	Variable length text field

Since columns 4 and 5 are mutually exclusive it is possible to optimize the mapping as follows:

#	Column name	Comment
1	Document number	Primary index
2	Node number	
3	Tag number (or tag name)	
4	Grove ordinal or -attribute ordinal	Attributes are flagged as negative ordinals
5	Placeholder flag	TRUE to give position in external grove
6	Content	Variable length text field

Normally, a row with placeholder flag = TRUE will have no contents, as it is used to mark the position of a node in the grove of which it is a part (external grove). However, if the contents of that node (internal grove) consist of exactly one element of text, then that text may be placed in the content of the placeholder row, saving an extra row. The database engine 20 can then implement an algorithm for reconstructing an XML document that has been parsed and mapped into a relational database as described above.

The algorithm involves a recursive procedure XML_OUT(doc_num,node_num) with two parameters:

1. The number for the document that is to be reconstructed.
2. A node number to operate on.

The routine is called with a document number and an initial node number of zero (root node), and performs the following steps:

1. Get all table rows with the specified document number and node number, and save to a list of rows.
2. Get the tag id from the row with placeholder flag = TRUE, translate it to a tag name if it is a number, and output a left angle bracket followed by the tag name.
3. Scan the list of rows for attribute rows (in the first mapping column $5 \neq 0$, in the second mapping column $4 < 0$). For each, translate the tag number into an

attribute name (if needed) and output the attribute name followed by an equal sign and followed by the row content. Separate the attributes by whitespace.

4. Discard all the attribute rows from the list.
5. Output a right angle bracket, ending the tag.
6. If the placeholder flag is TRUE, skip the next 3 steps.
7. Using the transform matrix, find the numbers of all the nodes that are children of the current node.
8. For each such node, get the row indexed by the document number, node number and placeholder flag of TRUE and add it to the list of rows.
9. Sort the list of rows by grove ordinal.
10. For each entry in the list, in order, do the following:

If this tag id is the same as the tag id from step 2, the row contains text contents, which are placed in the output, as is. Otherwise call XML_OUT with the document number and the node number from the row, to process an included tag.

11. Output a closing tag (left angle bracket, slash, tag name, right angle bracket).
12. Exit.

The routine can be optimized by first reading all the document rows in a suitable memory structure, and using its address instead of the document number.

The present invention thus allows XML documents to be parsed and stored in a relational database together with indexing information for each portion of the document content. This yields a number of important advantages. First, the documents can be efficiently searched without scanning the content of the entire document. In addition, the document or portions thereof can be accessed using a URI embedded in an address request without invoking any server side scripts. The XML database enables the creation of complex applications that are entirely browser based, using any XML compatible browser. Moreover, the XML documents or a portion or portions thereof can be extracted from the database and incorporated into other XML documents using the URI and the XML database can process and generate data interchange messages with a minimum of coding. If the interchange format can be used as part of the data model, no coding at all will be required.

The operation of the present invention as set forth above can be summarized by reference to the flowcharts of Figures 2 and 3. Figure 2 illustrates a process 200 for using the XML database of the present invention. The process is initiated by accessing (202) the XML database website. This may be implemented using an XML compatible browser. After the website is accessed, the client may establish (204) an XML database by registering a DTD with the database engine. This may be an interactive process involving a user friendly sequence of menus, screens and the like for prompting the user to enter the information required for the DTD definition. Once the XML database is established, any XML documents that comply with the DTD can be stored (206) in the database. The client may also provide (208) an application for using the XML database. Various kinds of applications may be provided in this regard for data mining, data analysis, customer service or the like. In addition, the application may be browser based. The client can then use (210) the application to access a document or portion thereof. For example, the client may use a browser to enter an address request that includes the URL of the database website, an identifier of the database to be accessed, and a URI for accessing particular portions of the database. In response to this request, the client will receive (212) the document or portions thereof. In accordance with the present invention, the client may optionally reuse (214) or interchange the document or portions thereof with other XML documents and applications.

Figure 3 illustrates a process 300 that may be executed in connection with database management. The process is initiated by establishing (302) a website with an XML database engine as described above. The engine is then utilized to store (304) a DTD for an XML database. The engine can subsequently validate (306) documents to be stored in the XML database by reference to the DTD. In this regard, only XML documents that are compliant with the DTD can be stored in the database and retrieved from it. If a document is validated, the document may be mapped (308) into a relational database as described above. When access to a stored document is desired, the engine receives (310) a URI, for example, embedded within an address request transmitted via an XML browser. In response to URI, the engine accesses (312) and reconstructs the requested document or portions thereof and transmits (314) the reconstructed document to the client via the network.

Although the present invention has been described in several embodiments, various changes and modifications may be suggested to one skilled in the art. It is intended that the present invention encompass such changes and modifications that fall within the scope of the appended claims.

What Is Claimed Is:

1. A method for use in accessing information in a network having an internet addressing system, comprising the steps of:

establishing a database of information objects accessible via said network;

receiving an information query from a client node embedded in an address request, said address request including at least a first portion identifying a network address of said server node and said information query, wherein said information query provides coded instructions for use in searching the database for one or more of said information objects; and

executing, at a server node of said network, said coded instructions of said information query to access said one or more information objects of said database or portions thereof, wherein said information objects are provided to said client node in response to said address request.

2. A method as set forth in Claim 1, wherein said step of establishing a database comprises registering an indexing definition comprising one of a Document Type Definition (DTD) and an XML Schema at said server node.

3. A method as set forth in Claim 2, wherein said indexing definition includes information about what tags may be used and how said tags may be nested for documents to be stored in and retrieved from said database.

4. A method as set forth in Claim 1, wherein said step of establishing comprises storing at least one XML document in said database.

5. A method as set forth in Claim 1, wherein said step of establishing comprises providing an XML interface for use in accessing said database using an XML browser.

6. A method as set forth in Claim 1, wherein said address request includes a URL for said server node and a Universal Resource Identifier (URI) for identifying said one or more information objects, and said step of receiving comprises identifying said URI.

7. A method as set forth in Claim 6, wherein said address request further includes information identifying said database.

8. A method as set forth in Claim 1, wherein said information query comprises an XPATH statement.

9. A method as set forth in Claim 1, wherein said step of executing said coded instructions comprises identifying one or more nodes associated with said one or more of said information objects.

10. A method as set forth in Claim 1, wherein said step of executing said coded instructions comprises identifying a first tag associated with said one or more of said information objects and utilizing a transformation matrix to identify related tags that are associated with said first tag via a family relationship.

11. A method for use in one of storing and accessing information in a network comprising the steps of:

establishing a database of information objects at a server node accessible via said network;

establishing an Extensible Markup Language (XML) interface in connection with said server node, whereby said XML interface is useful for accessing said database based on XML communications from client nodes;

receiving at said server node an XML communication from a client node; and

in response to said received XML communication, using said XML interface for one of storing and retrieving one or more information objects or portions thereof relative to said database.

12. A method as set forth in Claim 11, wherein said step of establishing a database comprises registering one of a Document Type Definition (DTD) and an XML Schema at said server node.

13. A method as set forth in Claim 11, wherein said step of establishing an XML interface comprises providing an XML interface module for receiving XML communications related to accessing said database.

14. A method as set forth in Claim 11, wherein said step of receiving comprises identifying an information query embedded in an address request submitted from said client node, wherein said address request includes a first portion identifying a network address of said server node and said information query.

15. A method as set forth in Claim 11, wherein said step of using said XML interface comprises executing coded instructions provided in an XPATH statement.

16. A method as set forth in Claim 11, wherein said step of using comprises parsing an XML document into a number of information objects and mapping said objects into a relational database.

17. A method as set forth in Claim 11, wherein said database is configured to store a parsed XML document in a relational database structure, and said step of using comprises reconstructing said parsed XML document or a portion or portions thereof.

18. A method as set forth in Claim 17, wherein said step of reconstructing comprises identifying a first tag associated with said parsed XML document and utilizing a transformation matrix to identify related tags that are associated with said first tag via a family relationship.

19. A method for use in one of storing and accessing information in a network, comprising the steps of:

establishing a database of information objects including at least one parent object and at least one child object wherein said parent object is associated with a parent tag system and said child object is associated with a child tag system and said child tag system is incorporated within said parent tag system;

defining a Boolean matrix reflecting parent and child relationships as between said information objects of said database; and

given information regarding a first one of said child object and said parent object, executing a transform operation using said Boolean matrix to identify a second one of said child object and said parent object different than said first one.

20. A method as set forth in Claim 19, wherein said step of establishing a database of information objects comprises mapping a parsed XML document into a relational database structure, wherein each of said parent object and said child object is a portion of said parsed XML document.

21. A method as set forth in Claim 20, further comprising the steps of using a family relationship between said parent object and said child object to reconstruct said parsed XML document or a portion or portions thereof.

22. A method as set forth in Claim 19, wherein said step of establishing a database of information objects comprises registering a DTD, wherein documents complying with said DTD can be stored in and retrieved from said database.

23. A method as set forth in Claim 19, further comprising the step of repeating said transform operation to identify further information objects having a family relationship with said first one of said child object and said parent object.

24. A method for use in storing information in a network, comprising the steps of:

receiving an Extensible Markup Language (XML) document;

parsing said XML document to create a number of information objects; and

mapping said XML document into a relational database by establishing, for each respective one of said information objects, a database entry including an information content of said respective one of said information objects and indexing information for use in identifying said respective one of said information objects.

25. A method as set forth in Claim 24, wherein said step of receiving comprises employing an XML interface to communicate with an XML browser.

26. A method as set forth in Claim 24, wherein said step of parsing comprises dividing said XML document into information segments and associating each of said segments with a tag of a family relationship tagging structure, wherein said tagging structure includes parent tags and child tags.

27. A method as set forth in Claim 25, wherein said step of mapping comprises establishing said indexing information such that said indexing information reflects a status of said tag relative to said family relationship tagging structure.

28. A method for use in accessing information in a network, comprising the steps of:

accessing database structure storing Extensible Markup Language (XML) documents, wherein a particular XML document is parsed into separate information objects and said objects are mapped into a relational database;

causing an algorithm to be executed to retrieve said separate information objects of said particular XML document and to reconstruct said particular XML document or portion thereof from said information objects; and

receiving said reconstructed particular XML document or portion thereof from said database structure.

29. A method as set forth in Claim 28, wherein said step of accessing comprises utilizing an XML compatible browser.

30. A method as set forth in Claim 28, wherein said step of causing an algorithm to be executed comprises entering an address request with an embedded Universal Resource Identifier (URI), wherein said address request comprises a single statement including a first portion for identifying a database server associated with said database structure and a second portion including said URI for use in identifying said separate information objects of said particular XML document.

31. A method as set forth in Claim 28, wherein said step of causing an algorithm to be executed comprises causing said algorithm to identify said separate information objects based on family relationships between said objects as defined by a tagging structure.

32. A method as set forth in Claim 28, further comprising the step of using said reconstructed document or portion thereof by one of reusing and interchanging associated data in a separate document or application.

33. An apparatus for use in one of establishing and using an Extensible Markup Language (XML) database, comprising: a server associated with a particular address of a network; an XML interface module, running on said server, for interfacing with an XML browser; an XML database associated with said server and configured to store at least one XML document; and a database engine operatively associated with said XML database for accessing said database in response to client instructions for storage and retrieval of said XML document.

34. An apparatus as set forth in Claim 33, wherein said server comprises an HTTP server.

35. An apparatus as set forth in Claim 33, wherein said XML interface module is operative for receiving a Universal Resource Identifier (URI) from said XML browser, wherein said URI includes one of a mode ordinal and a logical XPATH statement for identifying said XML document or a portion or portions thereof.

36. An apparatus as set forth in Claim 33, wherein said database is configured to store said XML document in parsed form in a relational database structure.

37. An apparatus as set forth in Claim 36, wherein said database engine is operative for reconstructing said parsed XML document using a family relationship between tags associated with segments of said parsed XML document, wherein information regarding said family relationship is indexed in said relational database structure.

38. An apparatus as set forth in Claim 33, further comprising memory structure configured for storing a Document Type Definition (DTD), wherein said DTD is used by said database engine to validate documents for storage in and retrieval from said database.

1/3

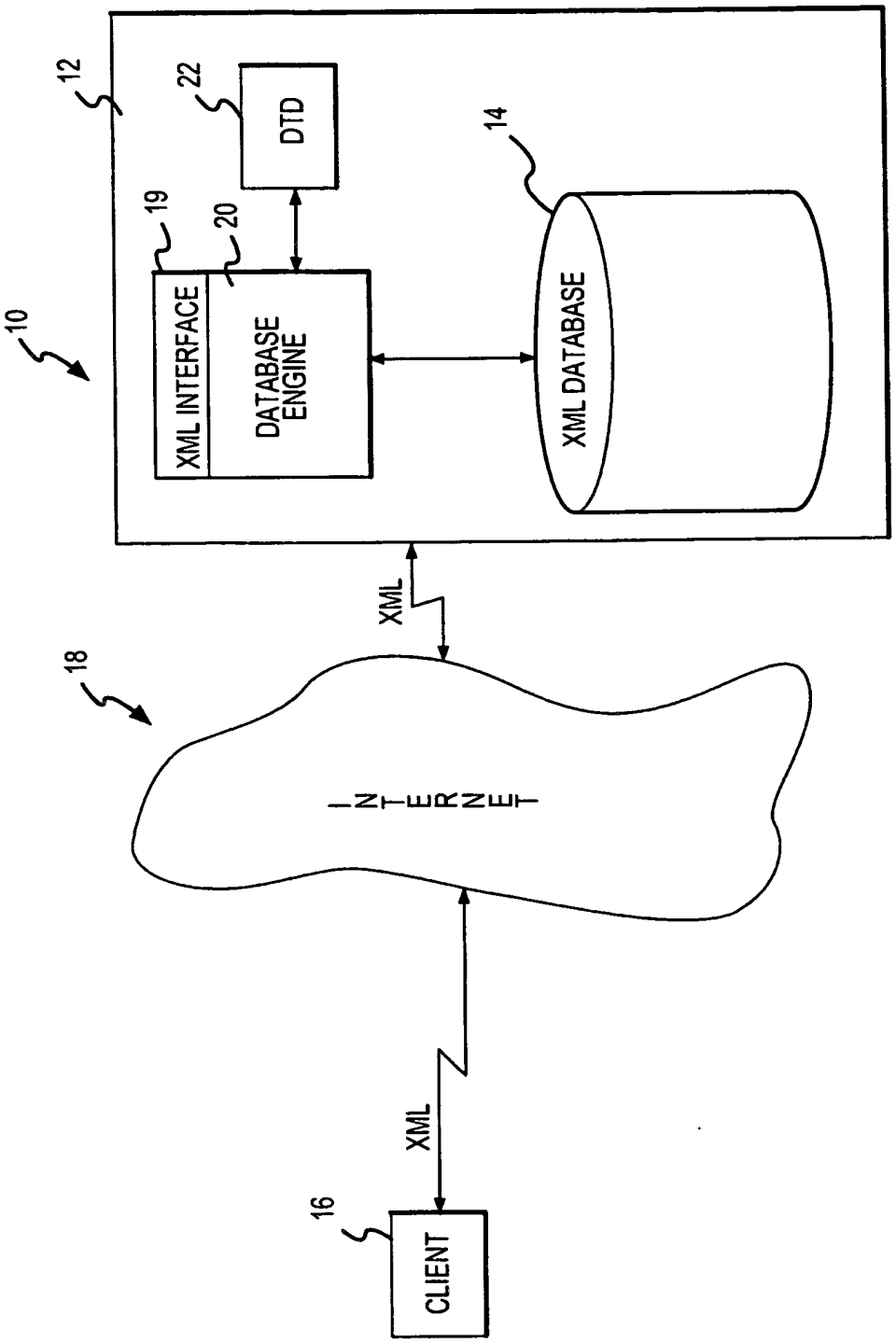

12
22
DTD
14
19
20
10
XML INTERFACE
DATABASE ENGINE
XML DATABASE
XML
18
INTERNET
XML
16
CLIENT

FIG.1

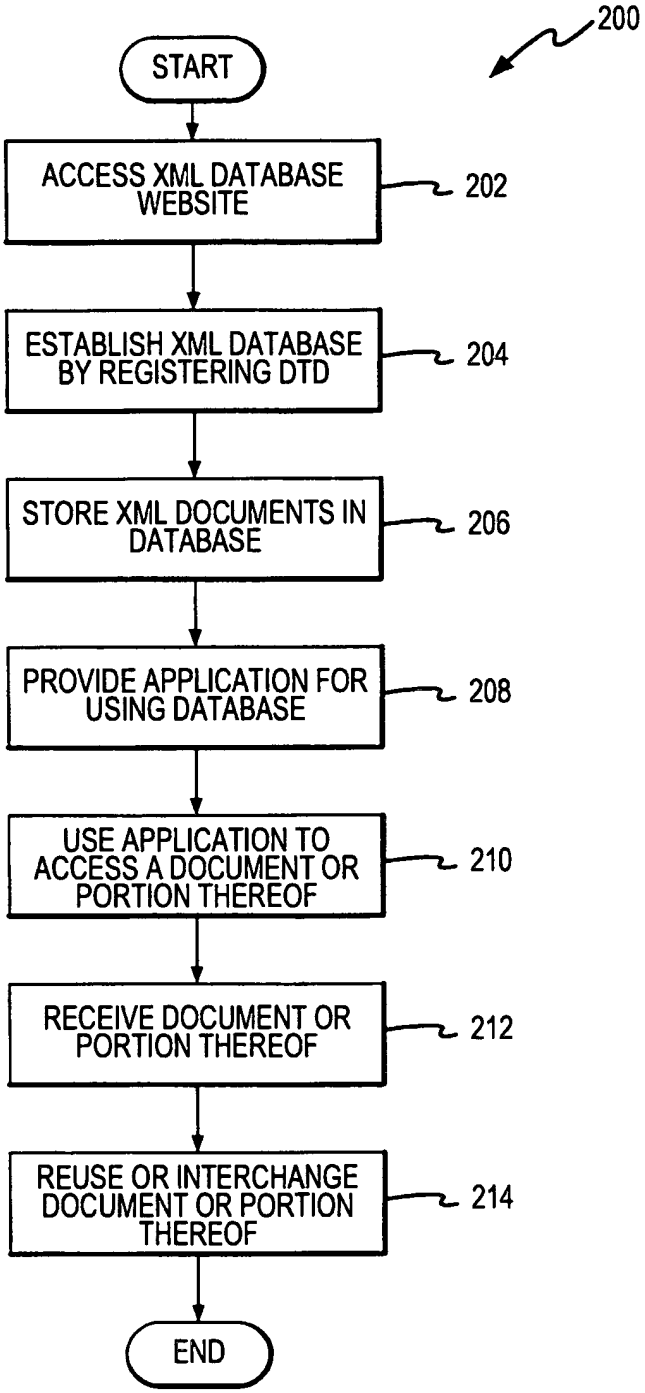

200
START
ACCESS XML DATABASE WEBSITE
202
ESTABLISH XML DATABASE BY REGISTERING DTD
204
STORE XML DOCUMENTS IN DATABASE
206
PROVIDE APPLICATION FOR USING DATABASE
208
USE APPLICATION TO ACCESS A DOCUMENT OR PORTION THEREOF
210
RECEIVE DOCUMENT OR PORTION THEREOF
212
REUSE OR INTERCHANGE DOCUMENT OR PORTION THEREOF
214
END

FIG.2

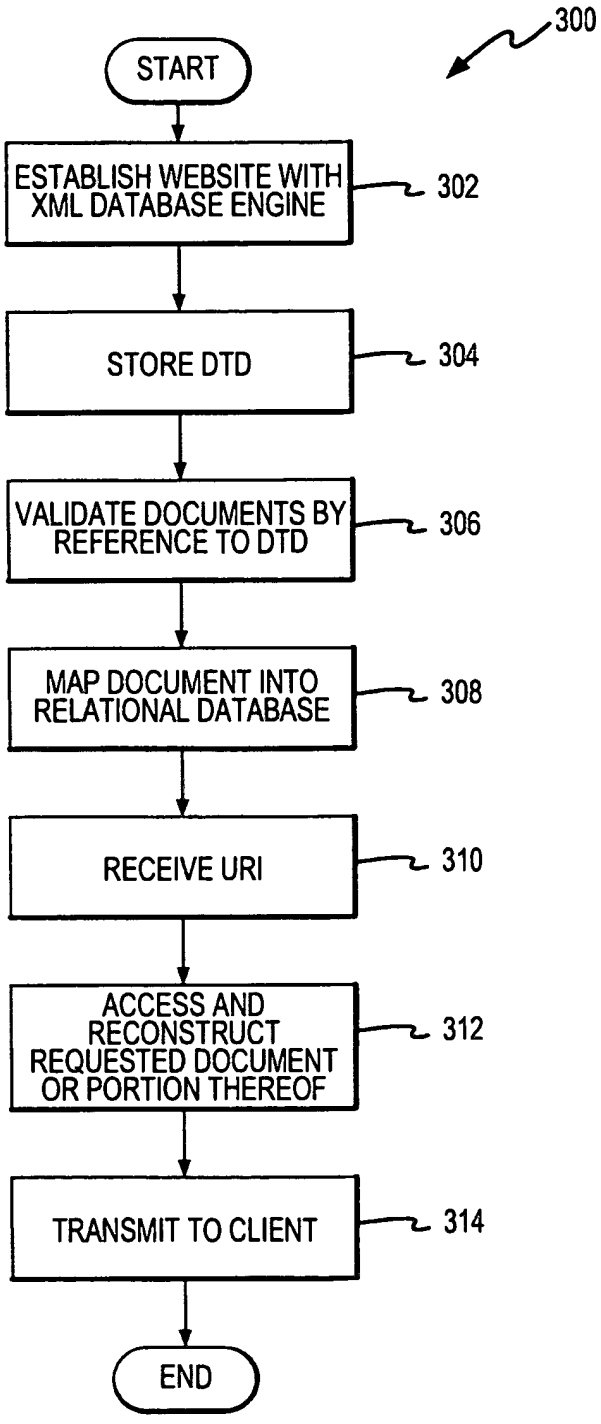

300
START
ESTABLISH WEBSITE WITH XML DATABASE ENGINE
302
STORE DTD
304
VALIDATE DOCUMENTS BY REFERENCE TO DTD
306
MAP DOCUMENT INTO RELATIONAL DATABASE
308
RECEIVE URI
310
ACCESS AND RECONSTRUCT REQUESTED DOCUMENT OR PORTION THEREOF
312
TRANSMIT TO CLIENT
314
END

FIG.3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/30020

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 17/30, 7/00, 17/00
US CL :707/10, 102, 103; 709/201, 203,
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 707/10, 102, 103; 709/201, 203,

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EAST, IEL, ACM

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X, P ---- Y,P	US 6,055,569 A (O'BRIEN et al) 25 April 2000, see the abstract, Figures 1-4.	1, 6-10 ------- 2-5,11-18,24-38
X --- Y	US 5,410,688 A (WILLIAMS et al) 25 April 1995, see the abstract.	19 --- 20-23
Y, P	US 6,012,098 A (BAYEH et al) 04 January 2000, see Figure 5, column 11, lines 3-19.	2, 3, 22
Y, P	US 6,167,564 A (FONTANA et al) 26 December 2000, see column 6, lines 63-66.	20, 21, 23, 28-32
Y	CIANCARINI et al, Managing Complex Documents over the WWW: a Case Study for XML, IEEE 1999, pages 629-638, see the abstract.	2 - 5 , 1 1 - 18,20,21,23-38

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 16 JANUARY 2001	Date of mailing of the international search report 15 FEB 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer UYEN LE Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)*